Universal Adaptor Signatures from Blackbox Multi-Party Computation

Michele Ciampi[1], Xiangyu Liu[23], Ioannis Tzannetos[24], and Vassilis Zikas[3]

[1] University of Edinburgh
[2] Purdue University
[3] Georgia Institute of Technology
[4] National Technical University of Athens
michele.ciampi@ed.ac.uk, liu3894@purdue.edu, itzannet@purdue.edu, vzikas@gatech.edu

Abstract. Adaptor signatures (AS) extend the functionality of traditional digital signatures by enabling the generation of a *pre-signature* tied to an instance of a hard NP relation, which can later be turned (adapted) into a full signature upon revealing a corresponding witness. The recent work by Liu et al. [ASIACRYPT 2024] devised a generic AS scheme that can be used for any NP relation—which here we will refer to as *universal adaptor signatures* scheme, in short UAS—from any one-way function. However, this generic construction depends on the Karp reduction to the Hamiltonian cycle problem, which adds significant overhead and hinders practical applicability.

In this work, we present an alternative approach to construct universal adaptor signature schemes relying on the *multi-party computation in the head* (MPCitH) paradigm. This overcomes the reliance on the costly Karp reduction, while inheriting the core property of the MPCitH—which makes it an invaluable tool in efficient cryptographic protocols—namely, that the construction is black-box with respect to the underlying cryptographic primitive (while it remains non-black-box in the relation being proven). Our framework simplifies the design of UAS and enhances their applicability across a wide range of decentralized applications, such as blockchain and privacy-preserving systems. Our results demonstrate that MPCitH-based UAS schemes offer strong security guarantees while making them a promising tool in the design of real-world cryptographic protocols.

1 Introduction

Blockchain technology was born out of a need for secure, transparent, and decentralized systems, with Bitcoin [18] serving as the first major application. Initially, blockchain's primary purpose was to enable peer-to-peer digital transactions without relying on a centralized authority. Over time, this innovation laid the foundation for decentralized finance (DeFi), a financial ecosystem built on blockchain, offering an alternative to traditional financial intermediaries. DeFi enables lending, borrowing, trading, and other financial services through smart

contracts, allowing users to maintain control over their assets without depending on centralized entities.

As DeFi rapidly grew, it gained significant adoption within and beyond the cryptocurrency space, attracting users, developers, and institutional players. The promise of financial inclusion, coupled with the potential to earn yields and participate in decentralized governance, spurred widespread interest. However, with this increase in adoption came notable challenges, particularly in terms of scalability. Popular blockchains like Ethereum struggled with network congestion, leading to high transaction fees and slower processing times. These limitations emphasized the critical need for blockchain systems to handle the growing demand efficiently. Scalability remains a central focus for the continued growth of DeFi and blockchain applications, with efforts directed toward reducing bottlenecks, increasing transaction throughput, and lowering fees to ensure that blockchain networks can support a global user base.

A method that has been proposed to address such scalability challenges is the use of adaptor signatures (AS) [19,2]. Adaptor signatures enable more efficient transaction processing by reducing the amount of data that needs to be stored and validated on-chain. This cryptographic technique allows participants to conditionally reveal certain information, such as secrets, only when specific conditions are met. By minimizing the need for complex scripts and offloading much of the computation, adaptor signatures help streamline transactions and reduce associated fees.

Adaptor signatures (AS) were introduced by Poelstra [19,20]. A first formal definition was given by Aumayr et al. [1,2]. It defined AS as an extended version of digital signatures which incorporates a cryptographic relation between a so-called *pre-signature* and a witness for an NP statement. In more detail, in an AS scheme, the signer produces a pre-signature tied to a hard relation R, and any party who knows the witness y to Y (where $(Y, y) \in R$) can transform (i.e., adapt) this pre-signature into a valid signature. Once the full signature is made public, the signer can extract the witness y from both the pre-signature and the full signature. Due to these features adaptor signatures can introduce a layer of fairness and trust between participants, which makes them particularly useful in scenarios like fair exchanges and cross-chain swaps.

The notion of AS also has the natural requirement that the adapted signature does not leak information about the witness. This guarantees that only those who generated the pre-signature will get access to the witness. Such a feature becomes particularly relevant in the blockchain context, in which for example an entity wants to sell the witness for Y, conditioned on being paid in some cryptocurrency. AS can be used to perform exactly this task atomically, by letting the buyer of y issue a pre-signature for a transaction that pays the seller. The seller, to get the money simply needs to complete the signature by adapting it. Upon generating the adapted signature, the seller will be rewarded, and the extraction property of the AS scheme guarantees that the buyer will learn the witness y. It is easy to see that in such a scenario, it would be preferable that the adapted signature (that will be posted on the blockchain) does not leak information about y, as

this would allow anyone participating in the blockchain protocol to learn y for free. This property was formalized as *witness hiding* in [17].

AS schemes can be applied in blockchain systems to reduce on-chain computation while maintaining security. Key applications include:

Atomic Swaps. Atomic swaps [20,7] allow two parties to exchange assets from different cryptocurrencies (e.g., Bitcoin and Ethereum) without intermediaries. Both parties lock their funds on-chain and exchange pre-signatures tied to a shared hard relation instance. Once the pre-signature is adapted by one party using their witness, the other party can extract the witness and adapt the second pre-signature, ensuring a fair exchange of assets without one party holding an unfair advantage.

Cross-Chain Atomic Swaps. Similar to atomic swaps, cross-chain swaps [12,6] involve two different blockchains, where assets from different chains are exchanged securely. Alice and Bob lock funds on their respective blockchains and exchange pre-signatures for transactions. The same instance Y and witness y are used across both chains, allowing each party to adapt their pre-signatures once they observe the other party's adapted signature, ensuring secure asset transfer across both blockchains.

Multi-Hop Payments. Multi-hop payments [7] enable multiple parties to route payments among themselves via a common intermediary, assuming they share a payment channel. While the original protocol [7] has each party sample a new instance-witness pair for every transaction, we simplify it by allowing all parties to use the same instance, provided that intermediate parties do not collude. Consider three parties: Alice, Bob, and Charlie where Alice wishes to pay cryptocurrency (say c) to Charlie. First, Alice and Bob lock funds on the blockchain within a payment channel as collateral, followed by Bob and Charlie. Charlie then randomly samples an instance-witness pair (Y, y) and forwards the instance Y to Alice and Bob. Alice initiates the transaction by creating a pre-signature $\tilde{\sigma}_A$ and sending it to Bob. Bob, in turn, generates his own pre-signature $\tilde{\sigma}_B$ and forwards it to Charlie. Once Charlie receives $\tilde{\sigma}_B$, he can convert it into a full signature σ_B by using the witness y, allowing him to finalize the transaction on the blockchain and receive the amount c. Simultaneously, Bob can derive the witness y from $\tilde{\sigma}_B$ and σ_B, adapt his pre-signature $\tilde{\sigma}_A$ into a full signature σ_A, and complete his transaction to obtain c.

Universal AS (UAS). While most previous works [2,24,7,23,15,21] focus on constructing AS schemes based on specific signatures schemes (such as ECDSA [2] and Schnorr [2,24]) and for script-related relations (like the public/secret key relation in signatures), only two works [4,17] have explored AS schemes for generic NP relations, which we refer to as *universal adaptor signature schemes (in short, UAS)*. Dai et al. [4] gave a generic construction of universal AS. However, their scheme does not provide witness hiding as the witness is fully exposed in the adapted signature itself. To address this issue, Liu et al. [17] proposed a construction that leverages the NP-completenes of the the Hamiltonian cycle problem. In particular, they creates an AS scheme which can be used for the case of Hamilto-

nian cycles. Since any NP relation can be reduced to that problem, this provides a universal AS scheme. However, when the target NP language we need for our application is not the language of Hamiltonian cycles—which will be the case in most, if not all, applications of universal AS—then before using the scheme of [17] a Karp reduction [14] needs to be used to reduce the statement at hand to an instance of the Hamilton cycle problem. This makes the above result of purely theoretical value, because the Karp reductions (in particular to Hamilton cycle) is a computationally expensive operation. Thus an question left open by [17] is whether we can avoid relying on the expensive Karp reduction, which takes us closer to a practical system.

In fact, Liu et al. suggested that a possible way to get such a result could be by using the multi-party computation in the head (MPCitH)paradigm [13] which has proven invaluable in the design of practical (zero-knowledge) proofs. However, as observed there, the techniques of [17] appear to be incompatible with MPCitH. The reason is that the scheme of [17] requires the existence of a sigma protocol in which the first round is independent of the witness (a property satisfied by the sigma-protocol for Hamiltonian cycles proposed in [3]). This feature is not satisfied by the sigma-protocol obtained via the MPCitH approach. Indeed, in this, the first round consists of the commitments of the parties' views that executed an MPC protocol to check the membership of an instance-witness pair in an NP relation. In particular the input of these parties corresponds to a share of the witness. Hence, the views depend on the witness itself, and as such the first round of the obtained sigma-protocol is not witness-independent. As such [17] left the following natural question remained open:

Question: *How to construct more efficient UAS schemes from the MPCitH paradigm?*

In this work, we solve this open problem by proposing a new approach to construct universal adaptor signatures from the MPCitH paradigm.

1.1 Our Contributions

In this work, we present a novel approach to construct universal adaptor signatures (UAS) from the *multi-party computation in the head* (MPCitH) paradigm [13]. Furthermore, we enhance the applicability across a wide range of decentralized applications, such as blockchain and privacy-preserving systems. Our results demonstrate that MPCitH-based AS schemes lead to more efficient protocols, making them a promising candidates for real-world cryptographic protocols.

1.2 Technical Overview

In this section we take a brief overview of our approach.

Generic Construction for NP Relations [17]. We start by recalling the generic construction proposed in [17]. The key insight is that the process used to

extract the witness of universal adaptor signatures (UAS) shares some similarities with the notion of *special soundness* of sigma protocols. Special soundness requires that a witness can be extracted from two valid transcripts (that refer to the same NP instance) with the same first-round message but different challenges (we refer to Section 2.5 for a formal definition of sigma protocols). In Liu et al.'s generic construction, a pre-signature for a message m with respect to an instance Y (a Hamiltonian graph) is of the form

$$\tilde{\sigma} = (\bar{\sigma}, cmt, ch = m_{\mathbf{0}}, rsp),$$

where $\bar{\sigma}$ is a regular signature on message (m, Y, cmt), and (cmt, ch, rsp) is a simulated transcript of Blum's sigma protocol w.r.t. instance Y for the Hamiltonian cycle problem [3,9], with $m_{\mathbf{0}}$ being an all-zero string. Moreover, Blum's sigma protocol has the property that one can simulate a transcript with the challenge 0^λ, and then use the same first-round message to reply to a different challenge if the witness for the statement being proven is known. Hence, with the knowledge of a witness y, the pre-signature can be adapted into a full signature

$$\sigma = (\bar{\sigma}, cmt, ch' = m, rsp'),$$

where (cmt, ch', rsp') is another valid transcript with challenge being the message m to be signed. By the special soundness of the sigma protocol, a witness y for the instance Y can be extracted from both the pre-signature $\tilde{\sigma}$ and the adapted signature σ.

MPC-in-the-Head Paradigm. Now we recall the multi-party computation in the head (MPCitH) paradigm [13], which transforms a secure n-party MPC protocol Π into a sigma protocol. Let R be an NP relation with Y an instance and y a corresponding witness such that $(Y, y) \in R$. Let f_R be the functionality such that

$$f_R(Y, y_1, ..., y_n) = 1 \text{ if and only if } (Y, \bigoplus_{i \in [n]} y_i) \in R.$$

Let Π be an MPC protocol securely realizing f_R. The high-level idea of the MPCitH paradigm is that, the prover, with public input Y and private input y, first shares y into n pieces of secrets $y_1, ..., y_n$, and then simulates "in the head" the running of Π with party P_i taking y_i as the local private input. The prover then commits the views of all parties, where the view View_i consists of the input and output of P_i, the randomness used for P_i, and the transcripts sent to/from P_i. Upon receiving commitments of all views, the verifier randomly samples two distinct indexes i, j, and asks the prover to open the corresponding views of P_i and P_j. The verifier then checks whether View_i and View_j are well-formed and consistent with each other. In this informal part of the paper, we will refer to this protocol with Π^{MPCitH}. Unfortunately, this protocol cannot be used in the approach of [17] we have just recalled due to the first round being dependent on the witness. For this reason, we will follow a different approach.

Our new approach. The high-level idea is as follows. Let (pk, sk) be a public-secret-key pair of a standard unforgeable signature scheme. The public-secret key of our adaptor signature corresponds exactly to (pk, sk).

To issue a pre-signature for an NP statement Y and message m, the signer generates a public-secret-key pair for a public-key encryption scheme (pk^e, sk^e), and sign (using the standard signature scheme) (m, Y, pk^e) thus obtaining $\bar{\sigma}$. The pre-signature is represented by the tuple

$$\tilde{\sigma} = (pk^e, sk^e, \bar{\sigma}).$$

To adapt the signature (i.e., to complete the signature) we require the adaptor to execute Π^{MPCitH} proving the knowledge of a witness for y, but with the following difference. In the first round the views of the MPC protocol are not committed but encrypted using the public-key pk^e. In summary, the adapted signature simply consists of Π^{MPCitH}, where the views in the first round are computed using a public-key encryption scheme instead of a commitment scheme. What we have just described would make the generation of the adapted signature an interactive protocol (given that Π^{MPCitH} is interactive). To make it non-interactive, we apply the Fiat-Shamir transform [8] thus obtaining a proof π, which represents our adapted signature

$$\sigma = (Y, pk^e, \bar{\sigma}, \pi).$$

We now argue that our scheme satisfies the most relevant properties required by a secure adaptor signature scheme.

- **Unforgeability.** This is inherited from the underlying signature scheme. Indeed, generating a standard signature requires generating a pre-signature and adapting the signature. This is clearly impossible unless sk is known, or the adversary breaks the unforgeability property of the standard signature scheme we use.
- **Witness extractability.** The extractabiltiy property requires that having the pre-signature $\tilde{\sigma}$ and the adapted signature σ with respect to a statement Y, it must be possible to extract the witness for Y. Note that the pre-signature contains the secret key sk^e of the encryption scheme used to generate the first round of Π^{MPCitH}. The extractor can use sk^e to decrypt the first round message of Π^{MPCitH} and extract all the shares contained in the view of the MPC parties, and finally reconstruct the witness. Arguing that the extraction is successful even when the adapted signature is generated by a malicious adversary, requires some care, and we refer the reader to the technical section of the paper for more details.
- **Witness hiding.** Witness hiding requires that by looking at the adapted signature σ (without the knowledge of the pre-signature $\tilde{\sigma}$), no adversary should be able to infer something about the witness for Y. We recall that the adapted signature σ simply corresponds to an application of the Fiat-Shamir transform on the protocol Π^{MPCitH}. This in a nutshell corresponds to a non-interactive zero-knowledge proof, which by definition hides the witness for the statement proven.

1.3 Roadmap

This paper is organized as follows. In Section 2, we present the preliminaries, including extractable commitments, the MPCitH paradigm, and the syntax and security notions of adaptor signatures. In Section 3 we show our main contribution: the generic construction of UAS from the MPCitH paradigm, along with its security proof. In Section 4 we discuss several issues related to the application of UAS on blockchains.

2 Preliminaries

We use $\lambda \in \mathbb{N}$ to denote the security parameter throughout this paper. For $\mu \in \mathbb{N}$, define $[\mu] := \{1, 2, ..., \mu\}$. Denote by $x := y$ the operation of assigning y to x. Denote by $x \xleftarrow{\$} \mathcal{S}$ the operation of sampling x uniformly at random from a set $\mathcal{S}$. For a distribution $\mathcal{D}$, denote by $x \leftarrow \mathcal{D}$ the operation of sampling x according to $\mathcal{D}$. For an algorithm $\mathcal{A}$, denote by $y \leftarrow \mathcal{A}(x; r)$, or simply $y \leftarrow \mathcal{A}(x)$, the operation of running $\mathcal{A}$ with input x and randomness r and assigning the output to y. For deterministic algorithms $\mathcal{A}$, we also write as $y := \mathcal{A}(x)$ or $y := \mathcal{A}(x; r)$. "PPT" is short for probabilistic polynomial-time.

Min-entropy. Let X a random variable (distribution) defined over $\mathcal{S}$. The min-entropy of X is defined as $\mathbf{H}_\infty(X) := -\log(\max_{s \in \mathcal{S}} \Pr[X = s])$.

2.1 Extractable Commitments

We recall the definition of extractable commitments in [22].

Definition 1 (Extractable Commitments). *An extractable commitment scheme* $\mathsf{eCOM} = (\mathsf{TdGen}, \mathsf{Com}, \mathsf{Ver}, \mathsf{Ext})$ *consists of the following four algorithms.*

- $(ck, td) \leftarrow \mathsf{TdGen}(1^\lambda)$. *The trapdoor key generation algorithm takes as input the security parameter* λ*, and outputs a commitment key* ck *and a trapdoor* td*.*
  We denote as $(ck, td) := \mathsf{TdGen}(1^\lambda; r)$ *if the randomness* r *is specified during trapdoor key generation.*
- $(c, d) \leftarrow \mathsf{Com}(ck, m)$. *The commitment algorithm takes as input* ck *and a message* m*, and outputs a commitment* c *and an opening* d*.*
- $0/1 \leftarrow \mathsf{Ver}(ck, c, m, d)$. *The verification algorithm takes as input* ck*,* c*,* m *and* d*, and outputs a bit.*
- $m' \leftarrow \mathsf{Ext}(td, c)$. *The extraction algorithm takes as input* td *and* c*, and outputs a message* m'*.*

Correctness*. For any* $(ck, td) \leftarrow \mathsf{Gen}(1^\lambda)$*, any message* m *and* $(c, d) \leftarrow \mathsf{Com}(ck, m)$*, it holds that* $\mathsf{Ver}(ck, c, m, d) = 1$*.*

If the trapdoor td is set to be empty, then we define a regular commitment scheme $\mathsf{Com} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver})$, where TdGen is replaced with the key generation algorithm $ck \leftarrow \mathsf{Gen}(1^\lambda)$.

Definition 2 (Extractability). *An extractable commitment scheme* eCOM *has extractability, if for any unbounded adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{ext}_{\mathsf{eCOM},\mathcal{A}}(\lambda) := \left|\Pr\left[\begin{array}{c}(ck,td) \leftarrow \mathsf{TdGen}(1^\lambda) \\ (c,m,d) \leftarrow \mathcal{A}(ck); m' \leftarrow \mathsf{Ext}(td,c)\end{array} : \begin{array}{c}\mathsf{Ver}(ck,c,m,d)=1 \\ \wedge\ m \neq m'\end{array}\right]\right|$$

is negligible over λ*.*

We say eCOM *has perfect extractability if* $\mathsf{Adv}^{ext}_{\mathsf{eCOM},\mathcal{A}}(\lambda) = 0$*.*

In our construction of adaptor signatures in Section 3, we require honest extractability, which guarantees perfect extractability against honest committers.

Definition 3 (Statistical Binding). *An extractable commitment scheme* eCOM *has statistical binding, if for any unbounded adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{binding}_{\mathsf{eCOM},\mathcal{A}}(\lambda) := \left|\Pr\left[\begin{array}{c}(ck,td) \leftarrow \mathsf{TdGen}(1^\lambda) \\ (c,m_0,m_1,d_0,d_1) \leftarrow \mathcal{A}(ck)\end{array} : \begin{array}{c}m_0 \neq m_1 \\ \wedge\ \mathsf{Ver}(ck,c,m_0,d_0)=1 \\ \wedge\ \mathsf{Ver}(ck,c,m_1,d_1)=1\end{array}\right]\right|$$

is negligible over λ*.*

We say eCOM *has perfect binding if* $\mathsf{Adv}^{binding}_{\mathsf{eCOM},\mathcal{A}}(\lambda) = 0$*.*

Definition 4 (Hiding of Extractable Commitments). *An extractable commitment scheme* eCOM *has hiding, if for any stateful PPT adversary* $\mathcal{A}$*, the advantage*

$$\begin{aligned}\mathsf{Adv}^{hiding}_{\mathsf{eCOM},\mathcal{A}}(\lambda) := \Bigg|&\Pr\left[\begin{array}{c}(ck,td) \leftarrow \mathsf{TdGen}(1^\lambda); (m_0,m_1,st) \leftarrow \mathcal{A}(ck) \\ (c,d) \leftarrow \mathsf{Com}(ck,m_0)\end{array} : \mathcal{A}(st,c)=1\right] \\ &- \Pr\left[\begin{array}{c}(ck,td) \leftarrow \mathsf{TdGen}(1^\lambda); (m_0,m_1,st) \leftarrow \mathcal{A}(ck) \\ (c,d) \leftarrow \mathsf{Com}(ck,m_1)\end{array} : \mathcal{A}(st,c)=1\right]\Bigg|\end{aligned}$$

is negligible over λ*.*

The properties of statistical binding and (computational) hiding for a regular commitment scheme COM can be defined similarly.

Definition 5 (κ-Min-Entropy of Commitments). *An extractable commitment scheme* eCOM *has* κ*-min-entropy, if for any* $(ck,td) \leftarrow \mathsf{TdGen}(1^\lambda)$*, any message* m*, we have* $\mathbf{H}_\infty(c) \geq \kappa$*, where* c *is computed as* $(c,d) \leftarrow \mathsf{Com}(ck,m)$*.*

Extractable commitments can be constructed from public-key encryption (PKE) schemes, where the public key serves as the commitment key, the secret key serves as the trapdoor, and the commitment is a ciphertext of a message with r—the randomness used in the encryption—being the opening. We refer to Appendix A for the definition of PKE and the transform we have just highlighted.

2.2 Signatures

Definition 6 (Signatures). *A signature scheme* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ *consists of the following three algorithms.*

- $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. *The key generation algorithm takes as input the security parameter* λ*, and outputs a public key* pk *and a secret key* sk*.*
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$. *The signing algorithm takes as input* sk *and a message* m*, and outputs a signature* σ*.*
- $0/1 \leftarrow \mathsf{Ver}(pk, m, \sigma)$. *The verification algorithm takes as input* pk*,* m*, and* σ*, and outputs a bit* b *indicating the validity of* σ *(w.r.t.* m*).*

Correctness*. For any* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, any message* m *and* $\sigma \leftarrow \mathsf{Sign}(sk, m)$*, it holds that* $\mathsf{Ver}(pk, m, \sigma) = 1$*.*

Definition 7 (Unforgeability of Signatures). *A signature scheme* SIG *is unforgeable under chosen message attacks (UF-CMA secure), if for any PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{uf}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{uf}_{\mathsf{SIG},\mathcal{A}}(\lambda) = 1]$$

is negligible over λ*, where the experiment* $\mathsf{Exp}^{uf}_{\mathsf{SIG},\mathcal{A}}(\lambda)$ *is defined in Fig. 1.*

$\mathsf{Exp}^{uf}_{\mathsf{SIG},\mathcal{A}}(\lambda)$:	$\textsc{Sign}(m)$:
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$; $\mathcal{S} := \varnothing$	$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}(\cdot)}(pk)$	$\mathcal{S} := \mathcal{S} \cup \{m\}$
	Return σ
Return $((m^* \notin \mathcal{S}) \wedge (\mathsf{Ver}(pk, m^*, \sigma^*)))$	

Fig. 1. The UF-CMA security experiment of signature scheme SIG.

2.3 NP Languages

Let $\{R_\lambda\} \subseteq (\{0,1\}^* \times \{0,1\}^*)_\lambda$ be a series of binary relations indexed by parameter λ. If λ is fixed then we simply denote R_λ as R. We call R an NP relation if there is an efficient algorithm to check whether $(Y, y) \in R$. The relation R defines an NP language $\mathcal{L}_R := \{Y \in \{0,1\}^* \mid \exists y \in \{0,1\}^* \ s.t. \ (Y, y) \in R\}$. We call Y the instance (not necessarily in $\mathcal{L}_R$), and y a witness of Y if $(Y, y) \in R$. Usually, there is an efficient sample algorithm that returns an instance-witness pair. Formally, $(Y, y) \leftarrow \mathsf{Sample}(R)$.

Definition 8 (Hard Relations). *A binary relation* R *is hard (one-way), if for any PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{ow}_{R,\mathcal{A}}(\lambda) := \Pr[(Y, y) \leftarrow \mathsf{Sample}(R); y' \leftarrow \mathcal{A}(R, Y) \ : \ (Y, y') \in R]$$

is negligible over λ*.*

2.4 (Universal) Adaptor Signatures

Definition 9 ((Universal) Adaptor Signatures [2,17]). *An adaptor signature scheme w.r.t. a relation R consists of seven algorithms* $\mathsf{AS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{pSign}, \mathsf{pVer}, \mathsf{Adapt}, \mathsf{Ext})$*, where the first three algorithms are defined as regular signatures (cf. Def. 6), and the last four are defined as follows.*

- $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$*. The pre-sign algorithm takes as input sk, m and an instance Y, and outputs a pre-signature $\tilde{\sigma}$.*
- $b \leftarrow \mathsf{pVer}(pk, m, Y, \tilde{\sigma})$*. The pre-verification algorithm takes as input pk, m, Y and $\tilde{\sigma}$, and outputs a bit indicating the validity of $\tilde{\sigma}$.*
- $\sigma \leftarrow \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)$*. The adaption algorithm takes as input pk, m, $\tilde{\sigma}$ and a witness y as input, and outputs an adapted signature σ.*
- $y/\perp \leftarrow \mathsf{Ext}(pk, m, Y, \tilde{\sigma}, \sigma)$*. The extraction algorithm takes as input pk, m, Y, $\tilde{\sigma}$ and σ, and outputs a witness y, or a failure symbol $\perp$.*

Except for the correctness as defined in Def. 6, we additionally require the pre-signature correctness, the adaption correctness, and the extraction correctness.

For any $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, any message m, any $(Y, y) \in \mathcal{R}$, any* $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$*,* $\sigma \leftarrow \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)$*, and* $y' \leftarrow \mathsf{Ext}(pk, m, Y, \tilde{\sigma}, \sigma)$*, it holds that*

1. *(Pre-signature correctness)* $\mathsf{pVer}(pk, m, Y, \tilde{\sigma}) = 1$*, and*
2. *(Adaption correctness)* $\mathsf{Ver}(pk, m, \sigma) = 1$*.*
3. *(Extraction correctness)* $(Y, y') \in R$*.*

We call an AS scheme a universal adaptor signature (UAS) scheme (denoted as UAS*), if the scheme is defined for generic NP relations.*

We require unforgeability, witness extractability, pre-signature adaptability, and witness hiding for the security of adaptor signatures.

Unforgeability. The adversary cannot generate a valid pre-signature or a regular signature on a fresh message.

Witness extractability. The signer can extract a witness from the pre-signature and the adapted signature.

Pre-signature adaptability. The receiver can adapt a valid pre-signature into a valid (full) signature with the knowledge of a witness.

Witness hiding The adapted signature leaks no additional information about the witness.

We formally define these security notions, following the works by Dai et al. [4] and Liu et al. [17].

Definition 10 (Unforgeability of Adaptor Signatures). *An adaptor signature scheme* AS *w.r.t. binary relation R is unforgeable under chosen message attacks (UF-CMA secure), if for any PPT adversary $\mathcal{A}$,* $\mathsf{Adv}^{uf}_{\mathsf{AS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{uf}_{\mathsf{AS},\mathcal{A}}(\lambda) = 1]$ *is negligible over λ, where the experiment* $\mathsf{Exp}^{uf}_{\mathsf{AS},\mathcal{A}}(\lambda)$ *is defined in Fig. 2.*

$\underline{\mathsf{Exp}^{uf}_{\mathsf{AS},\mathcal{A}}(\lambda)}$:	$\underline{\textsc{Sign}(m)}$:	$\underline{\textsc{NewY}()}$:
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$; $\mathcal{S} := \varnothing$; $\mathcal{T}[m] := \varnothing$	$\sigma \leftarrow \mathsf{Sign}(sk, m)$	$(Y, y) \leftarrow \mathsf{Sample}(R)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}(\cdot),\textsc{pSign}(\cdot,\cdot),\textsc{NewY}()}(pk)$	$\mathcal{S} := \mathcal{S} \cup \{m\}$	$\mathcal{Y} := \mathcal{Y} \cup \{Y\}$
	Return σ	Return Y
Return $(b_1 \wedge (b_{2,1} \vee b_{2,2}))$, where		
b_1: $\mathsf{Ver}(pk, m^*, \sigma^*) = 1 \ \wedge \ m^* \notin \mathcal{S}$	$\underline{\textsc{pSign}(m, Y)}$:	
$b_{2,1}$: $\mathcal{T}[m^*] = \varnothing$	$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$	
$b_{2,2}$: $\forall (Y, \tilde{\sigma}) \in \mathcal{T}[m^*] : Y \in \mathcal{Y}$	$\mathcal{T}[m] := \mathcal{T} \cup \{(Y, \tilde{\sigma})\}$	
	Return $\tilde{\sigma}$	

Fig. 2. The UF-CMA security experiment of adaptor signature scheme AS.

Definition 11 (Witness Extractability of Adaptor Signatures). *An adaptor signature scheme* AS *w.r.t. relation* R *is witness extractable, if for any PPT adversary* $\mathcal{A}$, $\mathsf{Adv}^{we}_{\mathsf{AS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{we}_{\mathsf{AS},\mathcal{A}}(\lambda) = 1]$ *is negligible over* λ, *where the experiment* $\mathsf{Exp}^{we}_{\mathsf{AS},\mathcal{A}}(\lambda)$ *is defined in Fig. 3.*

$\underline{\mathsf{Exp}^{we}_{\mathsf{AS},\mathcal{A}}(\lambda)}$:	$\underline{\textsc{Sign}(m)}$:
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$; $\mathcal{S} := \varnothing$; $\mathcal{T}[m] := \varnothing$	$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}(\cdot),\textsc{pSign}(\cdot,\cdot)}(pk)$	$\mathcal{S} := \mathcal{S} \cup \{m\}$
	Return σ
Return $(b_1 \wedge b_2)$, where	
b_1: $\mathsf{Ver}(pk, m^*, \sigma^*) = 1 \ \wedge \ m^* \notin \mathcal{S}$	$\underline{\textsc{pSign}(m, Y)}$:
b_2: $\forall (Y, \tilde{\sigma}) \in \mathcal{T}[m^*] : (Y, \mathsf{Ext}(pk, m^*, Y, \tilde{\sigma}, \sigma^*)) \notin R$	$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
// all $(Y, \tilde{\sigma})$ in the pre-sign list lead to a failed extraction	$\mathcal{T}[m] := \mathcal{T} \cup \{(Y, \tilde{\sigma})\}$
	Return $\tilde{\sigma}$

Fig. 3. The witness extractability experiment of adaptor signature scheme AS.

Definition 12 (Pre-signature Adaptability of Adaptor Signatures). *An adaptor signature scheme* AS *w.r.t. relation* R *has pre-signature adaptability, if for any public key* pk, *any message* m, *any* $(Y, y) \in R$ *and pre-signature* $\tilde{\sigma}$ *s.t.* $\mathsf{pVer}(pk, m, Y, \tilde{\sigma}) = 1$, *it holds that* $\mathsf{Ver}(pk, m, \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)) = 1$.

Definition 13 (Witness Hiding of Adaptor Signatures). *An adaptor signature scheme* AS *w.r.t. relation* R *is witness hiding, if there exists a simulator* Sim *such that, for any PPT adversary* $\mathcal{A}$,

$$\mathsf{Adv}^{wh}_{\mathsf{AS},\mathsf{Sim}\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{wh}_{\mathsf{AS},\mathsf{Sim},\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{wh}_{\mathsf{AS},\mathsf{Sim},\mathcal{A},1}(\lambda) = 1]|$$

is negligible over λ, *where the experiments* $\mathsf{Exp}^{wh}_{\mathsf{AS},\mathsf{Sim},\mathcal{A},b}(\lambda)$ $(b \in \{0,1\})$ *are defined in Fig. 4.*

$\underline{\mathsf{Exp}^{wh}_{\mathsf{AS},\mathsf{Sim},\mathcal{A},b}(\lambda)}$:	$\underline{\textsc{Chall}_0(pk, sk, m)}$	$\underline{\textsc{Chall}_1(pk, sk, m)}$
$(Y, y) \leftarrow \mathsf{Sample}(R)$	If $f_{\mathsf{AS}}(pk, sk) \neq 1$: Return $\perp$	If $f_{\mathsf{AS}}(pk, sk) \neq 1$: Return $\perp$
Return $\mathcal{A}^{\textsc{Chall}_b(\cdot,\cdot,\cdot)}(Y)$	// check the validity of (pk, sk)	// check the validity of (pk, sk)
	$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$	$\sigma \leftarrow \mathsf{Sim}(pk, sk, m, Y)$
	$\sigma \leftarrow \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)$	Return σ
	Return σ	

Fig. 4. The witness hiding experiments of adaptor signature scheme AS.

2.5 Sigma Protocols

Definition 14 (Sigma Protocols [5]). *A sigma protocol Σ for an NP relation R is a three-move interactive protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ working as follows.*

- $0/1 \leftarrow \langle \mathcal{P}(y), \mathcal{V} \rangle(Y)$. *Both the prover $\mathcal{P}$ and the verifer $\mathcal{V}$ share the common input Y, and $\mathcal{P}$ additionally takes y as a private input. After the interaction, $\mathcal{V}$ outputs a bit b for accept/reject, based on the transcript and Y.*
- *The transcript consists of three moves: the first move is a commitment cmt sending from $\mathcal{P}$ to $\mathcal{V}$. The second move is a challenge ch sending from $\mathcal{V}$ to $\mathcal{P}$, where ch is sampled from some finite set $\mathcal{CH}$. The third move is a response rsp sending from $\mathcal{P}$ to $\mathcal{V}$.*

We require the following properties for Σ.

- ***Completeness**. For all $(Y, y) \in R$, the verifier $\mathcal{V}$ outputs 1 after an honest running of the protocol, i.e., $1 \leftarrow \langle \mathcal{P}(y), \mathcal{V} \rangle(Y)$.*
- ***Special Soundness** There is a polynomial time extractor $\mathcal{E}$ such that given any pair of accepted transcripts (cmt, ch, rsp) and (cmt, ch', rsp') for any instance Y such that $ch \neq ch'$, the extractor $\mathcal{E}$ can extract a witness y such that $(Y, y) \in R$ except for a negligible probability.*
- ***Honest-Verifier Zero-Knowledge**. There is a PPT simulator Sim such that for all $(Y, y) \in R$, the output of $\mathsf{Sim}(Y)$ is distributed computationally close to the distribution of the transcript by an honest execution $\langle \mathcal{P}(y), \mathcal{V} \rangle(Y)$.*

Definition 15 (Public-Coin Sigma Protocols). *A sigma protocol Σ is public-coin if the challenge ch sent by the verifier $\mathcal{V}$ is uniformly sampled from the challenge space $\mathcal{CH}$.*

The well-known Fiat-Shamir transform [8] can be used to compile a public-coin sigma protocol into a non-interactive zero-knowledge arguement (NIZK) scheme in the random oracle model (ROM), by setting the challenge as $ch := \mathsf{H}(aux, cmt)$ and outputting $\pi = (cmt, rsp)$, where aux is some public information (e.g., the message to be signed when compiling a sigma protocol into a signature scheme in the ROM), and rsp is the response (the third move) of the sigma protocol. We omit the detailed description of the Fiat-Shamir transform here.

2.6 Multi-Party Computation

We recall the definition and security properties of MPC in [13].

Let $P_1, ..., P_n$ be n parties. All parties share a global input Y and each party P_i privately holds a share of secret y_i. We basically consider n-party functionality f_R w.r.t. an NP relation R which maps the input $(Y, y_1, ..., y_n)$ to an n-tuple of outputs.

The n-party protocol Π is specified via its next message function. More precisely, $\Pi(i, Y, y_i, r_i, (m_1, ..., m_r))$ returns the set of n messages sent by P_i in round $r+1$, given the global input Y, its local secret share y_i, its random coins r_i, and the messages $m_1, ..., m_r$ it received in the first r rounds. The output of Π also indicates that the protocol should terminate and return the final output of Π.

We use View_i to denote the view of P_i in the running of Π, which consists of y_i, r_i, and the messages P_i received. Note that if P_i is honest, then the messages P_i sent out and the final output of P_i are totally determined by View_i.

Definition 16 (Consistent Views). *A pair of views $\mathsf{View}_i, \mathsf{View}_j$ from parties P_i, P_j are consistent (w.r.t. the protocol Π and the global input Y), if the outgoing messages implicit in View_i are identical to the incoming messages reported in View_j and vice versa.*

Lemma 1 (Local v.s. Global Consistency [13]). *Let Π be an n-party protocol and Y be a global input. Let $\mathsf{View}_1, ..., \mathsf{View}_n$ be n views. All pairs of views $\mathsf{View}_i, \mathsf{View}_j$ are consistent (w.r.t. Π and Y) if and only if there exists an honest execution of Π with global input Y in which View_i is the view of P_i for all $i \in [n]$.*

Definition 17 (Correctness of Π). *We say Π realizes a deterministic n-party functionality $f(Y, y_1, ..., y_n)$ with perfect (resp., statistical) correctness if for all inputs $Y, y_1, ..., y_n$, the probability that the output of some player is different from the output of f is 0 (resp., negligible over λ), where the probability is over the independent choice of random coins $r_1, ..., r_n$.*

Definition 18 (t-Privacy of Π). *Let $1 \leq t < n$. We say that Π realizes f with prefect t-privacy if there exists a simulator Sim such that for any inputs $Y, y_1, ..., y_n$ and every set of corrupted parties $T \subset [n]$ s.t. $|T| \leq t$, the joint view $\mathsf{View}_T(Y, y_1, ..., y_n)$ of parties in T is distributed identically to the output of $\mathsf{Sim}(T, Y, (y_i)_{i \in T}, f_T(Y, y_1, ..., y_n))$. This definition can be relaxed to statistical or computational privacy defined similarly.*

2.7 MPC in the Head Paradigm

In this subsection we present the MPCitHead paradigm [13] that transfers an n-party MPC protocol Π into a public-coin sigma protocol Σ.

Let R be an NP relation, and f_R is a functionality corresponding to R such that

$$f_R(Y, y_1, ..., y_n) = 1 \text{ if and only if } (Y, \bigoplus_{i \in [n]} y_i) \in R.$$

Let COM be a commitment scheme with statistical binding and computational hiding, and $ck \leftarrow \mathsf{COM.Gen}(1^\lambda)$ be a commitment key known to both the prover $\mathcal{P}$ and the verifier $\mathcal{V}$.[5] Let Π be an $(n-1)$-private-n-party MPC protocol realizing f_R. We construct a sigma protocol $\Sigma^{\mathsf{COM}}_{\Pi,R,\ell} = \langle \mathcal{P}, \mathcal{V} \rangle$ for the NP relation R as follows. Note that we run $\ell = \Omega(\lambda)$ executions of Π in parallel to reduce the soundness error.

1. The setup algorithm returns $ck \leftarrow \mathsf{COM.Gen}(1^\lambda)$ as the common reference string.
2. For $j \in [\ell]$, $\mathcal{P}$ proceeds as follows:
   (a) Randomly sample $y_1^{(j)}, ..., y_{n-1}^{(j)}$, and set $y_n^{(j)} := y \bigoplus_{i \in [n-1]} y_i^{(j)}$.
   (b) Emulate "in their head" the execution of Π with inputs $(Y, y_1^{(j)}, ..., y_n^{(j)})$ and obtain views $\mathsf{View}_1^{(j)}, ..., \mathsf{View}_n^{(j)}$.
   (c) For $i \in [n]$, $(c_i^{(j)}, d_i^{(j)}) \leftarrow \mathsf{COM.Com}(ck, \mathsf{View}_i^{(j)})$.
3. $\mathcal{P}$ sends $(c_i^{(j)})_{i \in [n], j \in [\ell]}$ to $\mathcal{V}$.
4. $\mathcal{V}$ sends $(u^{(j)})_{j \in [\ell]}$ as the challenge to $\mathcal{P}$, where $u^{(j)} \in [n]$ is independently sampled at random for all $j \in [\ell]$.
5. $\mathcal{P}$ sends $(\mathsf{View}^{(j)}_{i \neq u^{(j)}}, d^{(j)}_{i \neq u^{(j)}})_{i \in [n], j \in [\ell]}$ to $\mathcal{V}$.
6. $\mathcal{V}$ accepts if and only if the followings hold for all $j \in [\ell]$.
   (a) For all $i \in [n] \backslash \{u^{(j)}\}$, $(\mathsf{View}_i^{(j)}, d_i^{(j)})$ is the correct opening of commitment $c_i^{(j)}$.
   (b) For all $i \in [n] \setminus \{u^{(j)}\}$, P_i finally output 1 (in the j-th execution).
   (c) For all distinct $i, i' \in [n] \setminus \{u^{(j)}\}$, $\mathsf{View}_i^{(j)}$ and $\mathsf{View}_{i'}^{(j)}$ are consistent with each other.

We have the following theorem, of which the proof is deferred to Appendix B.

Theorem 1. *Assume $n \geq 3$ be a constant. If Π realizes f_R with perfect correctness and either perfect, statistical, or computational $(n-1)$-privacy, and COM be a commitment protocol with statistical binding and computational hiding, then $\Sigma^{\mathsf{COM}}_{\Pi,R,\ell}$ described above is a public-coin sigma protocol for R.*

The 2-private 3-party GMW protocol [11,10] is prehaps the simplest instance that is appliable for the above MPCitH transformation. We refer to [13] for more MPC protocols in the literature.

The above described sigma protocol Σ (from the MPCitH paradigm) can further be transferred into a non-interactive zero-knowledge (NIZK) arguement scheme via the Fiat-Shamir transform [8] in the random oracle model (ROM). We omit the formal proof here.

[5] In our construction of universal adaptor signatures, the commitment scheme of the MPCitH paradigm is replaced with an extractable commitment scheme where the commitment key and the trapdoor are generated by the signer. Here we present the core construction of the MPCitH protocol as a warm-up of our construction in Section 3, and omit the generation of the commitment key.

3 Universal Adaptor Signatures from the MPCitH Paradigm

In this section we present our generic construction of universal adaptor signatures (UAS) from the MPCitH paradigm. Let R be a hard relation, and f is a functionality corresponding to R such that

$$f_R(Y, y_1, ..., y_n) = 1 \text{ if and only if } (Y, \bigoplus_{i\in[n]} y_i) \in R.$$

Let SIG be a signature scheme, eCOM be an extractable commitment scheme as defined in Section 2.1, Π be an n-party MPC protocol securly realizing the functionality f_R, $\ell = \Omega(\lambda)$, and $\mathsf{H} : \{0,1\}^* \to [n]^\ell$ be a random oracle. Our universal adaptor signature scheme UAS is constructed as follows.

- $\mathsf{Gen}(1^\lambda)$. The key generation algorithm returns $(pk, sk) \leftarrow \mathsf{SIG.Gen}(1^\lambda)$.
- $\mathsf{pSign}(sk, m, Y)$. On inputs sk, m and instance Y, the pre-signing algorithm does the following.
  1. $(ck, td) \leftarrow \mathsf{eCOM.TdGen}(1^\lambda; r)$, where r is the randomness uniformly sampled according to the specification of the tradpoor key generation algorithm of eCOM.
  2. $\bar{\sigma} \leftarrow \mathsf{SIG.Sign}(sk, (m, Y, ck))$.
  3. Return $\tilde{\sigma} := (r, \bar{\sigma})$.
- $\mathsf{pVer}(pk, m, Y, \tilde{\sigma})$. On inputs pk, m, Y, and $\tilde{\sigma} = (r, \bar{\sigma})$, the pre-verification algorithm does as follows.
  1. Compute $(ck, td) := \mathsf{eCOM.TdGen}(1^\lambda; r)$.
  2. Return $\mathsf{SIG.Ver}(pk, (m, Y, ck), \bar{\sigma})$.
- $\mathsf{Adapt}(pk, m, \tilde{\sigma}, y)$. On inputs pk, m, $\tilde{\sigma} = (r, \bar{\sigma})$ and witness y, the adaption algorithm does the following.
  - Compute $(ck, td) := \mathsf{eCOM.TdGen}(1^\lambda; r)$.
  - Generate a zero-knowledge proof $\pi = (cmt, rsp)$ for instance Y through the MPCitH paradigm and the Fiat-Shamir transform. In more detail, π is computed as follows:
    1. For $j \in [\ell]$, randomly sample $y_1^{(j)}, ..., y_{n-1}^{(j)}$, and set $y_n^{(j)} := y \bigoplus_{i\in[n-1]} y_i^{(j)}$.
    2. For $j \in [\ell]$, emulate the execution of Π with inputs $(Y, y_1^{(j)}, ..., y_n^{(j)})$ and obtain views $\mathsf{View}_1^{(j)}, ..., \mathsf{View}_n^{(j)}$.
    3. Set $cmt := (c_i^{(j)})_{i\in[n], j\in[\ell]}$, where $(c_i^{(j)}, d_i^{(j)}) \leftarrow \mathsf{eCOM.Com}(ck, \mathsf{View}_i^{(j)})$ for $i \in [n]$ and $j \in [\ell]$.
    4. Compute $\mathsf{H}(m, Y, ck, \bar{\sigma}, cmt) = (u^{(1)}, ..., u^{(\ell)})$.
    5. Set $rsp := (\mathsf{View}^{(j)}_{i\neq u^{(j)}}, d^{(j)}_{i\neq u^{(j)}})_{j\in[\ell]}$.
  - Return $\sigma := (Y, ck, \bar{\sigma}, \pi := (cmt, rsp))$.
- $\mathsf{Sign}(sk, m)$. On inputs sk and message m, the signing algorithm does the following.
  1. Sample a fresh instance-witness pair via $(Y, y) \leftarrow \mathsf{Sample}(R)$.
  2. $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$.

3. Return $\sigma \leftarrow \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)$.

– $\mathsf{Ver}(pk, m, \sigma)$. On inputs pk, m, and $\sigma = (Y, ck, \bar{\sigma}, \pi)$, the verification algorithm returns 1 if the followings hold and 0 otherwise.
  1. $\mathsf{SIG.Ver}(pk, (m, Y, ck), \bar{\sigma}) = 1$.
  2. π is a valid zero-knowledge proof, where the verification in proceeds as follows.
     (a) Parse $\pi = (cmt, rsp) = \left((c_i^{(j)})_{i\in[n],j\in[\ell]}, (\mathsf{View}_i^{(j)}, d_i^{(j)})_{i\in G^{(j)},j\in[\ell]}\right)$ with $G^{(j)} \subset [n]$ and $|G^{(j)}| = n - 1$.
     (b) Compute $\mathsf{H}(m, Y, ck, \bar{\sigma}, cmt) = (u^{(1)}, ..., u^{(\ell)})$.
     (c) Output 1 if and only if the following hold for all $j \in [\ell]$:
        i. $G^{(j)} = [n] \setminus \{u^{(j)}\}$;
        ii. $\mathsf{eCOM.Ver}(ck, c_i^{(j)}, \mathsf{View}_i^{(j)}, d_i^{(j)}) = 1$ for $i \in G^{(j)}$ (all openings of the commitments are correct).
        iii. For $i \in G^{(j)}$, party P_i finally outputs 1 (in the j-th execution).
        iv. Every pair of distinct views in $(\mathsf{View}_i^{(j)})_{i\in G^{(j)}}$ are consistent.

– $\mathsf{Ext}(pk, m, Y, \tilde{\sigma}, \sigma)$. On inputs pk, m, Y, $\tilde{\sigma} = (r, \bar{\sigma})$ and $\sigma = (Y', ck', \bar{\sigma}', \pi)$, the adaption algorithm proceeds as follows.
  1. Compute $\mathsf{eCOM.TdGen}(1^\lambda; r) = (ck, td)$ and $\mathsf{H}(m, Y, ck, \bar{\sigma}, cmt) = (u^{(1)}, ..., u^{(\ell)})$.
  2. If $(Y' \neq Y) \vee (ck' \neq ck)$, return $\perp$.
  3. If $\mathsf{Ver}(pk, m, \sigma) = 0$, return $\perp$.
  4. Parse $\pi = (cmt, rsp)$ and $cmt = (c_i^{(j)})_{i\in[n],j\in[\ell]}$. For $j \in [\ell]$ and $i \in [n]$, $\mathsf{View}_i^{(j)} \leftarrow \mathsf{eCOM.Ext}(td, c_i^{(j)})$.
  5. Let $y_i^{(j)}$ be the private input of $\mathsf{View}_i^{(j)}$. If there exist $j^* \in [\ell]$ such that $(Y, \bigoplus_{i\in[n]} y_i^{(j^*)}) \in R$, return $y := \bigoplus_{i\in[n]} y_i^{(j^*)}$. Otherwise, return $\perp$.

Theorem 2. *If* SIG *is an UF-CMA-secure signature scheme,* eCOM *is an extractable commitment scheme with extractability, statistical binding, computational hiding, and κ-min-entropy ($\kappa = \Omega(\lambda)$), R is a hard relation, Π is an n-party MPC protocol realizing f_R with $(n-1)$-privacy, and* H *works as a (programmable) random oracle, then* UAS *constructed above is a universal adaptor signature scheme with UF-CMA security, witness extractability, pre-signature adaptability, and witness hiding.*

Proof. The correctness of UAS is guaranteed by the correctness of SIG, the verification correctness and extractability of eCOM, and the correctness of Π.

Now we prove the security of UAS.

Witness extractability. Let $\mathcal{A}$ be an adversary against the witness extractability of UAS, and (m^*, σ^*) be $\mathcal{A}$'s final output, where $\sigma^* = (Y^*, ck^*, \bar{\sigma}^*, \pi^*)$. Recall that for $\mathcal{A}$ to win in the witness extractability experiment (cf. Def. 11), it must hold that

1. $\mathsf{SIG.Ver}(pk, (m^*, Y^*, ck^*), \bar{\sigma}^*) = 1$, π^* is a zero-knowledge proof for instance Y^* that the verifier accepts, and $\mathcal{A}$ never asks $\textsc{Sign}(m^*)$.
2. The witness extraction fails for all $(Y, \tilde{\sigma} = (r, \bar{\sigma}) \in \mathcal{T}[m^*])$.

We analyze $\mathcal{A}$'s advantage in two cases.

- **Case 1**. There exists no item $(Y, \tilde{\sigma} = (r, \bar{\sigma})) \in \mathcal{T}[m^*]$ with $\mathsf{eCOM.TdGen}(1^\lambda; r) = (ck, td)$ such that, $Y = Y^*$ and $ck = ck^*$.
- **Case 2**. There exist an item $(Y, \tilde{\sigma} = (r, \bar{\sigma})) \in \mathcal{T}[m^*]$ with $\mathsf{eCOM.TdGen}(1^\lambda; r) = (ck, td)$ such that, $Y = Y^*$ and $ck = ck^*$.

Recall that $\mathcal{A}$ never queries $\textsc{Sign}(m^*)$, and $\bar{\sigma}^*$ is a valid signature w.r.t. (m^*, Y^*, ck^*). If Case 1 happens, then we can construct a reduction algorithm $\mathcal{A}'$ that breaks the UF-CMA security of the underlying signature scheme SIG. We omit the details of the reduction here.

Now consider $\mathcal{A}$'s advantage in Case 2. Let Q_{H} be the total number of hash queries by $\mathcal{A}$. We first analyze the subcase where $\mathcal{A}$ has queried $\mathsf{H}(q^*) = \mathsf{H}(m^*, Y^*, ck^*, \bar{\sigma}^*, (c_i^{(j)})_{i\in[n],j\in[\ell]})$ before outputting $(m^*, \sigma^* = (Y^*, ck^*, \bar{\sigma}^*, \pi^*))$, where $\pi^* = \left((c_i^{(j)})_{i\in[n],j\in[\ell]}, (\mathsf{View}_i^{(j)}, d_i^{(j)})_{i\in G^{(j)},j\in[\ell]}\right)$.

We call $q = (m, Y, ck, \bar{\sigma}, cmt = (c_i^{(j)})_{i\in[n],j\in[\ell]})$ a *good* query, if there exists an item $(Y, \tilde{\sigma} = (r, \bar{\sigma})) \in \mathcal{T}[m]$ such that $\mathsf{eCOM.TdGen}(1^\lambda; r) = (ck, td)$. It is easy to see that in Case 2, the query q^* that related to $\mathcal{A}$'s final forgery is a good query.

For good query $q = (m, Y, ck, \bar{\sigma}, cmt = (c_i^{(j)})_{i\in[n],j\in[\ell]})$, define the following two events:

- Fail: the event that there exists no $j \in [\ell]$ such that $(Y, \bigoplus_{i\in n} y_i^{(j)}) \in R$, where $\overline{\mathsf{View}}_i^{(j)} \leftarrow \mathsf{eCOM.Ext}(td, c_i^{(j)})$, and $y_i^{(j)}$ is the private input of $\overline{\mathsf{View}}_i^{(j)}$ for $i \in [n]$ and $j \in [\ell]$. If Fail happens to a good query q, we also denote as $\mathsf{Fail}(q) = 1$.
- Pass: the event that there exists $\bar{\sigma}'$ and $rsp = (\mathsf{View}_i^{(j)}, d_i^{(j)})_{i\in G^{(j)},j\in[\ell]}$ such that $(m, \sigma = (Y, ck, \bar{\sigma}', cmt, rsp))$ passes the verification of UAS. If Pass happens to a good query q, we also denote as $\mathsf{Pass}(q) = 1$.

Given a good query $q = (m, Y, ck, \bar{\sigma}, cmt = (c_i^{(j)})_{i\in[n],j\in[\ell]})$, we analyze the probability that $\mathsf{Fail}(q) = 1$ and $\mathsf{Pass}(q) = 1$ in Case 2. Let $(Y, \tilde{\sigma} = (r, \bar{\sigma})) \in \mathcal{T}[m]$ and $\mathsf{eCOM.TdGen}(1^\lambda; r) = (ck, td)$. Recall that eCOM has extractability and statistical binding. Therefore, for $i \in [n]$ and $j \in [\ell]$, except for a negligible binding error, the commitment $c_i^{(j)}$ can only be opened to $\overline{\mathsf{View}}_i^{(j)} \leftarrow \mathsf{eCOM.Ext}(td, c_i^{(j)})$. By Lemma 1, if Fail happens, then there exist a group of indexes $(\hat{i}^{(1)}, ..., \hat{i}^{(\ell)}) \in [n]^\ell$ such that for any $j \in [\ell]$, either $\overline{\mathsf{View}}_{\hat{i}^{(j)}}^{(j)}$ exposes an output of 0 for party $P_{\hat{i}^{(j)}}$, or $\overline{\mathsf{View}}_{\hat{i}^{(j)}}^{(j)}$ is inconsistent with some other view $\overline{\mathsf{View}}_{i\neq\hat{i}^{(j)}}^{(j)}$.

Let $\mathsf{H}(q) = (u^{(1)}, ..., u^{(\ell)}) \in [n]^\ell$, and let $(\hat{i}^{(1)}, ..., \hat{i}^{(\ell)}) \in [n]^\ell$ be defined as above. For a message-signature pair $(m, \sigma = (Y, ck, \bar{\sigma}', cmt = (c_i^{(j)})_{i\in[n],j\in[\ell]}, rsp = (\mathsf{View}_i^{(j)}, d_i^{(j)})_{i\in G^{(j)},j\in[\ell]}))$ that passes the verification, it must hold that

1. $G^{(j)} = [n] \setminus \{u^{(j)}\}$;
2. For $i \in G^{(j)}$: $\mathsf{eCOM.Ver}(ck, c_i^{(j)}, \mathsf{View}_i^{(j)}, d_i^{(j)}) = 1$, and $\mathsf{View}_i^{(j)} = \overline{\mathsf{View}}_i^{(j)}$.

3. For $i \in G^{(j)}$, party P_i finally outputs 1 (in the j-th execution).
4. Every pair of distinct views in $(\mathsf{View}_i^{(j)})_{i \in G^{(j)}}$ are consistent.

Therefore, for a good query q, if Fail happens, then Pass happens only if $(u^{(1)}, ..., u^{(\ell)}) = (\hat{i}^{(1)}, ..., \hat{i}^{(\ell)})$. Since H is a random oracle and $\mathsf{H}(q)$ remains undefined until $\mathsf{H}(q)$ is invoked by $\mathcal{A}$, we have that

$$\Pr[\mathsf{Pass}(q) \wedge \mathsf{Fail}(q) \wedge \text{Case 2}] \leq \Pr[\mathsf{Pass}(q) \mid (\mathsf{Fail}(q) \wedge \text{Case 2})] \leq 1/n^{\ell}.$$

Let $(m^*, \sigma^* = (Y^*, ck^*, \bar{\sigma}^*, \pi^*))$ be $\mathcal{A}$'s final forgery. If $\mathcal{A}$ successes in Case 2, then there must exist a related good query $\mathsf{H}(q^*) = \mathsf{H}(m^*, Y^*, ck^*, \bar{\sigma}^*, cmt = c_i^{(j)})_{i \in [n], j \in [\ell]})$ such that $\mathsf{Pass}(q^*) = 1$ and $\mathsf{Fail}(q^*) = 1$. There are in total Q_{H} random oracle queries by $\mathcal{A}$. Therefore, the probability that $\mathcal{A}$ successes is bounded by Q_{H}/n^{ℓ}.

Then we consider the subcase of Case 2 where $\mathcal{A}$ has never queried $\mathsf{H}(q^*) = \mathsf{H}(m^*, Y^*, ck^*, \bar{\sigma}^*, (c_i^{(j)})_{i \in [n], j \in [\ell]})$. Similar to the analysis above, it is easy to see that $(\mathsf{Fail}(q^*) = 1 \wedge \mathsf{Pass}(q^*) = 1)$ cannot happen unless $\mathcal{A}$ successfully guess the output of $\mathsf{H}(q^*)$, which happens with probability $1/n^{\ell}$.

In summary, there exist an PPT adversary $\mathcal{A}'$ and a negligible function $negl(\cdot)$ such that

$$\mathsf{Adv}^{we}_{\mathsf{UAS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{uf}_{\mathsf{SIG},\mathcal{A}'}(\lambda) + \frac{Q_{\mathsf{H}} + 1}{n^{\ell}} + negl(\lambda),$$

where Q_{H} is the total number of random oracle queries by $\mathcal{A}$. This concludes the proof of witness extractability.

UF-CMA security. let $\mathcal{A}$ be an adversary against the UF-CMA security of UAS, and $(m^*, \sigma^* = (Y^*, ck^*, \bar{\sigma}^*, \pi^*))$ be $\mathcal{A}$'s final forgery in the unforgeability experiment (cf. Def. 10). Recall that for $\mathcal{A}$ to win, it must hold that

1. $\mathsf{SIG.Ver}(pk, (m^*, ck^*, Y^*), \bar{\sigma}^*) = 1$, and π^* is a valid zero-knowledge proof for instance Y^*.
2. $\mathcal{A}$ never queries $\textsc{Sign}(m^*)$, and
3. (a) either $\mathcal{T}[m^*] = \varnothing$ (i.e., $\mathcal{A}$ never asks $\textsc{pSign}(m^*, Y)$ for any Y), or
   (b) for all $(Y, \tilde{\sigma} = (r, \bar{\sigma})) \in \mathcal{T}[m^*]$, it holds that $Y \in \mathcal{Y}$ (i.e., $\mathcal{A}$ only queries $\textsc{pSign}(m^*, Y)$ for Y whose witness is unknown to it).

We first analyze the case $1 \wedge 2 \wedge (a)$. It is easy to see that in this case, $\mathcal{A}$ does not get any signature on message $(m^*, \cdot, \cdot)$ during either pre-signing queries or signing queries. Therefore, we can construct a reduction algorithm $\mathcal{A}_1$ that breaks the UF-CMA security of SIG.

Then we analyze the case $1 \wedge 2 \wedge (b)$. We further divide it into the following two subcases.

(i) For all $(Y, \tilde{\sigma} = (r, \bar{\sigma})) \in \mathcal{T}[m^*]$ and $\mathsf{eCOM.TdGen}(1^{\lambda}; r) = (ck, td)$, it holds that $(Y, ck) \neq (Y^*, ck^*)$.
This means that $\mathcal{A}$ does not obtain a signature on message (m^*, Y^*, ck^*) during either pre-signing queries or signing queries. Consequently, $\mathcal{A}$ breaks the UF-CMA security of the underlying SIG.

(ii) There exists $(Y, \tilde{\sigma} = (r, \bar{\sigma}))) \in \mathcal{T}[m^*]$ and $\mathsf{eCOM.TdGen}(1^\lambda; r) = (ck, td)$, such that $(Y, ck) = (Y^*, ck^*)$.
This means that $\mathcal{A}$ adapts some pre-signature to a valid full signature without knowing a witness. We analyze this case by showing an algorithm $\mathcal{A}_2$ such that, once the event $(1 \wedge 2 \wedge (b) \wedge (ii))$ happens, $\mathcal{A}_2$ is able to break the one-wayness of R.

$\mathcal{A}_2$ first receives an instance $\hat{Y}$ from its own challenger, and then simulates the experiment to $\mathcal{A}$ as follows.

1. $\mathcal{A}_2$ randomly selects $\eta \leftarrow [Q]$ with Q the number of queries to NewY() by $\mathcal{A}$. Then $\mathcal{A}_2$ generates $(pk, sk) \leftarrow \mathsf{SIG.Gen}(1^\lambda)$ and sends pk to $\mathcal{A}$.
2. $\mathcal{A}_2$ simulates the signing oracle Sign and the pre-signing oracle pSign as normal.
3. $\mathcal{A}_2$ simulates the new instance oracle NewY() as normal, except that $\mathcal{A}_2$ returns $\hat{Y}$ upon receiving the η-th query to NewY().
4. $\mathcal{A}_2$ simulates the random oracle H as normal. In more details, $\mathcal{A}_2$ maintains a table $\mathcal{H}$ which is initialized to be empty. Upon receiving a query $\mathsf{H}(q)$, if $\mathcal{H}[q]$ has not been defined, then $\mathcal{A}_2$ randomly samples $ch \leftarrow [n]^\ell$ and define $\mathcal{H}[q] := ch$. Finally $\mathcal{A}_2$ returns $\mathcal{H}[q]$.
5. At last $\mathcal{A}$ outputs $(m^*, \sigma^* = (Y^*, ck^*, \bar{\sigma}^*, \pi^* = (cmt^*, rsp^*)))$. If $Y^* \neq \hat{Y}$ then $\mathcal{A}_2$ aborts immediately. Otherwise, $\mathcal{A}_2$ checks if $(1 \wedge 2 \wedge (b) \wedge (ii))$ holds. If so, $\mathcal{A}_2$ retrieves $(Y^*, \tilde{\sigma} = (r, \bar{\sigma})) \in \mathcal{T}[m^*]$ such that $\mathsf{eCOM.TdGen}(1^\lambda; r) = (\tilde{ck^*}, td^*)$. Let $cmt^* = (c_i^{(j)})_{i \in [n], j \in [\ell]}$. For $j \in [\ell]$ and $i \in [n]$, $\mathcal{A}_2$ extracts the views $\mathsf{View}_i^{(j)} \leftarrow \mathsf{eCOM.Ext}(td^*, c_i^{(j)})$ and hence the private inputs $y_i^{(j)}$. If there exists $j^* \in [\ell]$ such that $(Y^*, \bigoplus_{i \in [n]} y_i^{(j^*)}) \in R$, then $\mathcal{A}_2$ outputs $y := \bigoplus_{i \in [n]} y_i^{(j^*)}$. Otherwise, $\mathcal{A}_2$ outputs $\perp$.

Similar to the analysis of Case 2 in the proof of witness extractability above, we know that for every good query $q = (m, Y, ck, \bar{\sigma}, cmt = (c_i^{(j)})_{i \in [n], j \in [\ell]})$, $\mathcal{A}_2$ successes in extracting a witness for Y except for a negligible probability $1/n^\ell$. Let Q_H be the total number of queries to the random oracle $\mathsf{H}(\cdot)$. Since the query q^* related to $\mathcal{A}$'s final forgery is good in case $(1 \wedge 2 \wedge (b) \wedge (ii))$, $\mathcal{A}_2$ is able to extract a witness of Y^* except for a probability $(Q_\mathsf{H} + 1)/n^\ell$.

In $\mathcal{A}_2$'s simulation, the challenge instance $\hat{Y}$ is randomly embedded into one query during the simulation of NewY(), and hence $Y^* = \hat{Y}$ holds with probability $1/Q$, where Q is the number of queries to NewY(). Therefore, if $\mathcal{A}$ breaks the unforgeability of UAS in case $(1 \wedge 2 \wedge (b) \wedge (ii))$ with probability ϵ, then $\mathcal{A}_2$ is able to break the one-wayness of the hard relation R with probability at least $\frac{\epsilon - (Q_\mathsf{H}+1)/n^\ell}{Q}$.

In summary, there exist PPT adversaries $\mathcal{A}_1, \mathcal{A}_2$, and a negligible function $negl(\cdot)$ such that

$$\mathsf{Adv}^{uf}_{\mathsf{UAS},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{uf}_{\mathsf{SIG},\mathcal{A}_1}(\lambda) + Q \cdot \mathsf{Adv}^{ow}_{R,\mathcal{A}_2}(\lambda) + \frac{Q_\mathsf{H} + 1}{n^\ell} + negl(\lambda),$$

which finishes the proof of unforgeability.

Pre-signature adaptability. Recall that a pre-signature is of the form $\tilde{\sigma} = (r, \bar{\sigma})$, where $\mathsf{eCOM.TdGen}(1^\lambda; r) = (ck, td)$ and $\bar{\sigma} \leftarrow \mathsf{SIG.Sign}(sk, (m, Y, ck))$. To adapt it to a full signature, it is only required to attach a zero-knowledge proof π^* for instance Y^*, using ck as the commitment key. The pre-signature adaptability is directly implied by the correctness of Π and the correctness of eCOM.

Witness hiding. To prove the witness hiding property of UAS, we need to design a simulator Sim which can simulate an adapted signature given (m, Y), which is indistinguishable from a signature generated by the pre-sign-and-adaption paradigm with the knowledge of y.

We design Sim similarly to the signing algorithm, with the only difference being that Sim generates a zero-knowledge proof π for instance Y without knowing a witness y as follows (which, is similar to the zero-knowledge simulator in the proof of Theorem 1 but works in the random oracle model).

In more details, Sim works as follows.

1. $(ck, td) \leftarrow \mathsf{eCOM.TdGen}(1^\lambda; r)$, where r is the randomness uniformly sampled according to the specification of eCOM.
2. $\bar{\sigma} \leftarrow \mathsf{SIG.Sign}(sk, (m, Y, ck))$.
3. For all $j \in [\ell]$, independently sample $u^{(j)} \leftarrow [n]$ at random.
4. For all $j \in [\ell]$, randomly sample $y_i^{(j)}$ for $i \in [n] \setminus \{u^{(j)}\}$, and invoke the simulator of the MPC protocol Π to obtain views $(\mathsf{View}_i^{(j)})_{i \in [n] \setminus \{u^{(j)}\}}$.
5. For all $j \in [\ell]$ and $i \in [n] \setminus \{u^{(j)}\}$, $(c_i^{(j)}, d_i^{(j)}) \leftarrow \mathsf{eCOM.Com}(ck, \mathsf{View}_i^{(j)})$.
6. For all $j \in [\ell]$, $(c_{u^{(j)}}^{(j)}, d_{u^{(j)}}^{(j)}) \leftarrow \mathsf{eCOM.Com}(ck, \mathbf{0})$ (where $\mathbf{0}$ denotes an all-zero bit string).
7. Let $cmt := (c_i^{(j)})_{i \in [n], j \in [\ell]}$. Reprogram H such that $\mathsf{H}(m, Y, ck, \bar{\sigma}, cmt) = (u^1, ..., u^{(\ell)})$. If the item already exists in $\mathcal{H}$, then abort and return $\perp$.
8. Finally, return the simulated signature $\sigma := (Y, ck, \bar{\sigma}, \pi := (cmt, rsp))$, where $rsp := \left((\mathsf{View}_i^{(j)})_{i \neq u^{(j)}}, (d_i^{(j)})_{i \neq u^{(j)}} \right)_{j \in [\ell]}$.

We first argue that the simulated transcript distributes computationally close to the transcript by an honest execution, if the reprogramming does not fail.

- The challenge $(u^{(1)}, ..., u^{(\ell)})$ simulated by Sim distributes identically to the honestly generated one since it is uniformly sampled.
- The private inputs in the response rsp simulated by Sim distribute identically to the honestly generated ones, since they are all independent and random strings.
- For every $j \in [\ell]$, the $n-1$ views revealed in the response distribute identically (resp., statistically/computationally close) to the honestly generate ones, due to the $(n-1)$-privacy of the MPC protocol Π.

– For every $j \in [\ell]$, all commitments $c_i^{(j)}$ for $i \in [n] \setminus \{u^{(j)}\}$ are distributed identically. Moreover, the commitment $c_{u^{(j)}}^{(j)}$ will not be revealed in the transcript, and hence $c_{u^{(j)}}^{(j)}$ in the simulation distributes computationally close to the one honestly generated, due to the hiding of eCOM.

Therefore, the simulated signature is distributed computationally close to the honestly generate one.

It is left to prove that the reprogramming fails with a negligible probability $Q_{\mathsf{H}}/2^{\kappa n \ell}$ with Q_{H} the number of hash queries by $\mathcal{A}$, which is guaranteed by the κ-min-entropy of eCOM. As a result, the witness hiding of UAS holds. □

4 Adaptor Signatures on Blockchains

While adaptor signatures (AS) offer an innovative approach to secure transactions and privacy-preserving protocols in decentralized applications, several technical challenges must be addressed during their deployment. These challenges primarily relate to security, efficiency, and the coordination of protocol components. Additionally, shifting some computations to off-chain approaches inherently raises valid concerns that need to be carefully managed.

4.1 Secure Transmission of Pre-Signatures

In the deployment of adaptor signatures, pre-signatures must be exchanged between participants (e.g., Alice and Bob in a cross-chain swap). Ensuring the secure transmission of these pre-signatures is crucial, as any interception could expose sensitive data. Pre-signatures contain critical cryptographic information and must be transmitted over secure channels, such as encrypted communication protocols (e.g., TLS), to prevent man-in-the-middle attacks. If we further assume a public key infrastrcture, the presignature should be encrypted under the public key of the witness holder, prior to transmission.

In the context of AS, if a malicious actor intercepts the message containing the pre-signature, even if they alter the data, no asset loss will occur. This is because the pre-signature verification algorithm will detect any modification, causing the procedure to abort. The worst-case scenario arises when a malicious actor learns the witness by possessing both the pre-signature and the adapted signature. In this case, without paying for the information, an adversary could exploit an insecure transmission channel to deduce the witness. Therefore, ensuring secure communication channels is paramount to protecting sensitive information and maintaining the integrity of the AS process.

4.2 Integrity of Pre-Signature Adaptation

A core component of adaptor signatures (AS) involves adapting pre-signatures into full signatures upon revealing a witness (e.g., a private key or secret). Ensuring the integrity of this adaptation process is essential. Any vulnerability in

the adaptation could allow an adversary to forge signatures. Careful implementation of cryptographic primitives and validation of the adaptation is required to avoid security breaches. The associated risks of forgery become greater if the AS algorithm necessitates a trusted setup, as this creates additional dependencies on the trustworthiness of the setup phase. If the setup is compromised or if the trusted entity that performed the setup, behaves maliciously, it could lead to significant security vulnerabilities, allowing unauthorized parties to forge signatures or otherwise undermine the integrity of the AS framework. Therefore, in cases involving a trusted setup, it is crucial that the inputs used during the setup are destroyed after their purpose has been fulfilled to mitigate the risk of exploitation.

4.3 Collusion and Replay Attacks

In scenarios where multiple parties are involved, the potential for collusion exists. An adversary may attempt to collaborate with one or more parties to exploit the adaptor signature mechanism, leading to unauthorized access to assets or information. Additionally, replay attacks may occur if a pre-signature is intercepted and reused in a different context. To mitigate these risks, it is imperative to implement robust protocols for nonce generation and transaction identifiers to ensure that each signature operation is unique and cannot be replayed.

The risk of replay attacks is heightened in account-based models, such as Ethereum, compared to the UTXO (Unspent Transaction Output) model, where transactions are directly tied to input UTXOs. In Ethereum's account-based model, smart contracts can be invoked multiple times, even if it was not the original intention, leading to unintended consequences based on the protocol's design. This vulnerability has made Ethereum susceptible to replay attacks, where a transaction could be executed again, potentially allowing adversaries to exploit the system and access assets without authorization. However, replay attacks can be mitigated if the smart contract implementing the protocol incorporates mechanisms to identify and prevent such invocations, ensuring that each transaction is processed uniquely and as intended.

4.4 Computational Complexity

The computational overhead associated with generating and adapting adaptor signatures can pose challenges, particularly in high-frequency trading environments or scenarios requiring rapid transaction throughput. As the number of participants increases, the complexity of managing the pre-signature and adaptation processes can lead to performance bottlenecks. Optimizing these processes while maintaining security is critical to achieving practical implementations of adaptor signatures (AS) in real-world applications.

To effectively scale up the use of AS, it is essential to identify every aspect of the framework that can be further optimized. Examining the algorithms and protocols used for signature generation and adaptation can reveal opportunities for efficiency improvements. In our specific case, the owner of the witness (e.g.,

the secret in an AS) can precompute the MPC-in-the-Head while the sender is transmitting the pre-signature to them. This concurrent precomputation could further enhance the system's efficiency by overlapping processes that would typically be sequential, thus reducing the time needed for signature adaptation and overall transaction finalization. By streamlining these processes and minimizing computational requirements, the AS framework can be better suited for high-demand applications and broader adoption in various sectors.

4.5 Further Considerations

In exploring the potential of adaptor signatures, several important considerations arise. Regarding Bitcoin compatibility, our current work necessitates a Turing-complete blockchain scripting language, such as Solidity for Ethereum, to validate the adapted signatures within smart contracts, which will result in significant costs of validation (transaction fees). On the other hand, Bitcoin's limited transaction scripting interface [16] is not capable of validating our scheme. Consequently, a potential avenue for future research is to find a method for constructing adaptor signatures derived from the MPC-in-the-head paradigm that can be validated on the Bitcoin network. Additionally, while GAS2 [4] provides a framework for creating adaptor signatures for strongly random-self reducible NP relations, it offers unlinkability—ensuring that the adapted signature is indistinguishable from a normally generated signature, regardless of the instance Y. However, it requires the strong random self-reducibility of the underlying relation, which confines the instantiations of GAS2 to standard number-theoretic problems, such as DL, RSA, and LWE. Our current construction does not support this unlinkability property, and thus a future research direction will involve developing unlinkable adaptor signatures based on our existing framework.

Acknowledgments

Michele Ciampi and Xiangyu Liu were partially funded by Sunday Group, Inc. Ioannis Tzannetos was funded by AnalytiXIN. Vassilis Zikas was funded in part by NSF grant No. 2055599 and Sunday Group, Inc.

References

1. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized bitcoin-compatible channels. IACR Cryptol. ePrint Arch. p. 476 (2020)
2. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized channels from limited blockchain scripts and adaptor signatures. In: ASIACRYPT 2021. vol. 13091, pp. 635–664. Springer (2021)
3. Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians. vol. 1, p. 2. Citeseer (1986)
4. Dai, W., Okamoto, T., Yamamoto, G.: Stronger security and generic constructions for adaptor signatures. In: INDOCRYPT 2022, Kolkata, India, December 11-14, 2022, Proceedings. vol. 13774, pp. 52–77. Springer (2022)

5. Dao, Q., Grubbs, P.: Spartan and bulletproofs are simulation-extractable (for free!). In: Advances in Cryptology - EUROCRYPT 2023, Lyon, France, April 23-27, 2023, Proceedings, Part II. vol. 14005, pp. 531–562. Springer (2023)
6. Deshpande, A., Herlihy, M.: Privacy-preserving cross-chain atomic swaps. In: International conference on financial cryptography and data security. pp. 540–549. Springer (2020)
7. Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. In: ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II. vol. 12309, pp. 378–397. Springer (2020)
8. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings. vol. 263, pp. 186–194. Springer (1986)
9. Goldreich, O.: The Foundations of Cryptography - Volume 1: Basic Techniques. Cambridge University Press (2001)
10. Goldreich, O.: The Foundations of Cryptography - Volume 2: Basic Applications. Cambridge University Press (2004)
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA. pp. 218–229. ACM (1987)
12. Gugger, J.: Bitcoin-monero cross-chain atomic swap. Cryptology ePrint Archive (2020)
13. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007. pp. 21–30. ACM (2007)
14. Karp, R.M.: Reducibility among combinatorial problems. In: Proceedings of a symposium on the Complexity of Computer Computations, 1972. pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972)
15. Klamti, J.B., Hasan, M.A.: Post-quantum two-party adaptor signature based on coding theory. Cryptogr. **6**(1), 6 (2022)
16. Liao, G., Wang, T., Yang, Q., Xia, Y., Shi, L., Zhao, X., Wu, X., Zhang, S., Chan, A., Yuen, R.: Programming on bitcoin: A survey of layer 1 and layer 2 technologies in bitcoin ecosystem. arXiv preprint arXiv:2409.19622 (2024)
17. Liu, X., Ioannis, T., Zikas, V.: Adaptor signatures: New security definition and A generic construction for NP relations. IACR Cryptol. ePrint Arch. p. 1051 (2024)
18. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Satoshi Nakamoto (2008)
19. Poelstra, A.: Scriptless scripts. Presentation Slides (3 2017), https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf, accessed on May 2024
20. Poelstra, A.: Mimblewimble and scriptless scripts. Presentation Slides (1 2018), https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2018-01-10-rwc/slides.pdf, accessed on May 2024
21. Renan, F., Kutas, P.: Sqiasignhd: Sqisignhd adaptor signature. IACR Cryptol. ePrint Arch. p. 561 (2024)
22. Santis, A.D., Crescenzo, G.D., Persiano, G.: Necessary and sufficient assumptions for non-iterative zero-knowledge proofs of knowledge for all NP relations. In: ICALP 2000. vol. 1853, pp. 451–462. Springer (2000)

23. Tairi, E., Moreno-Sanchez, P., Maffei, M.: Post-quantum adaptor signature for privacy-preserving off-chain payments. In: FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II. vol. 12675, pp. 131–150. Springer (2021)
24. Tu, B., Zhang, M., Yu, C.: Efficient ecdsa-based adaptor signature for batched atomic swaps. In: ISC 2022, Bali, Indonesia, December 18-22, 2022, Proceedings. vol. 13640, pp. 175–193. Springer (2022)

A Extractable Commitments from Public-Key Encryption

A.1 Public-Key Encryption

Definition 19 (Public-Key Encryption). *A public-key encryption (PKE) scheme* $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of the following there algorithms.*

- $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. *The key generation algorithm takes as input the security parameter* λ*, and outputs a public key* pk *and a secret key* sk*.*
- $ct \leftarrow \mathsf{Enc}(pk, m)$. *The (probabilistic) encryption algorithm takes as input* pk *and a message* m*, and outputs a ciphertext* ct*.*
  We denote as $ct := \mathsf{Enc}(pk, m; r)$ *if the randomness* r *is specified in encryption.*
- $m' \leftarrow \mathsf{Dec}(sk, ct)$. *The decryption algorithm takes as input* sk *and a ciphertext* ct*, and outputs a message* m'*.*

Correctness*. For any* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, any message* m *and* $c \leftarrow \mathsf{Enc}(pk, m)$*, it holds that* $\mathsf{Dec}(sk, ct) = m$*.*

Definition 20 (CPA Security of PKE). *A PKE scheme* PKE *is indistinguishable under chosen-plaintaxt attacks (CPA-secure), if for any stateful PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{cpa}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{matrix} (pk, sk) \leftarrow \mathsf{Gen}(1^\lambda); (m_0, m_1, st) \leftarrow \mathcal{A}(ck) \\ ct \leftarrow \mathsf{Enc}(pk, m_0) \end{matrix} \;:\; \mathcal{A}(st, ct) = 1 \right] \right.$$
$$\left. - \Pr\left[\begin{matrix} (pk, sk) \leftarrow \mathsf{Gen}(1^\lambda); (m_0, m_1, st) \leftarrow \mathcal{A}(ck) \\ ct \leftarrow \mathsf{Enc}(pk, m_1) \end{matrix} \;:\; \mathcal{A}(st, ct) = 1 \right] \right|$$

is negligible over λ*.*

Definition 21 (κ-Min-Entropy of Ciphertexts). *A PKE scheme* PKE *has* κ*-min-entropy, if for any* $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$*, any message* m*, we have* $\mathbf{H}_\infty(ct) \geq \kappa$*, where* ct *is computed as* $ct \leftarrow \mathsf{Enc}(ck, m)$*.*

A.2 Construction of Extractable Commitments from PKE

A PKE scheme PKE directly implies an extractable commitment scheme eCOM where the public key serves as the commitment key, the secret key serves as the trapdoor, and the commitment is a ciphertext of a message with r—the randomness used in the encryption—being the opening. It is easy to see that the extractable commitment scheme has perfect binding (due to the perfect correctness of PKE) and hiding (due to the CPA security of PKE).

More formally, let $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CPA-secure PKE scheme, we construct a extractable commitment scheme $\mathsf{eCOM} = (\mathsf{TdGen}, \mathsf{Com}, \mathsf{Ver}, \mathsf{Ext})$ as follows.

- $(ck, td) \leftarrow \mathsf{TdGen}(1^\lambda)$. The trapdoor key generation algorithm returns $ck := pk$ and $td := sk$, where $(pk, sk) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$.
- $(c, d) \leftarrow \mathsf{Com}(ck, m)$. The commitment algorithm returns $(c, d) := (ct, r)$, where $ct := \mathsf{PKE.Enc}(ck, m; r)$ and r is a fresh randomness sampled according to the specification of the encryption scheme.
- $0/1 \leftarrow \mathsf{Ver}(ck, c, m, d)$. The verification algorithm returns 1 if $c = \mathsf{PKE.Enc}(ck, m; d)$ and 0 otherwise.
- $m' \leftarrow \mathsf{Ext}(td, c)$. The extraction algorithm returns $\mathsf{PKE.Dec}(td, c)$.

Theorem 3. *If* PKE *is a CPA-secure PKE scheme with perfect correctness, then* eCOM *constructed above is a secure extractable commitment scheme with perfect binding and computational hiding. Moreover, if* PKE *has κ-min-entropy, then* eCOM *also has κ-min-entropy.*

Proof. The correctness of eCOM follows from the fact that $\mathsf{PKE.Enc}(pk, m; r)$ is a deterministic function, and the extractability of eCOM is derived from the correctness of PKE.

Now we prove the binding property of eCOM based on the correctness of PKE. Recall that for every ck generated from $(ck = pk, sk) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$, an encryption of m will always be decrypted into m. If there is an all-powerful adversary that given ck, outputs a collusion (c, m_0, m_1, d_0, d_1), then we have $\mathsf{PKE.Enc}(pk, m_0; d_0) = c = \mathsf{PKE.Enc}(pk, m_1; d_1)$ and $m_0 \neq m_1$, which is conflict to the perfect correctness of PKE since the decryption of c, an encryption of m_0, may lead to a distinct message m_1.

It has been left to prove the hiding property of COM, i.e., for any messages m_0 and m_1 chosen by the adversary, a commitment of m_0 (which is an encryption of m_0) is indistinguishable from a commitment of m_1 (which is an encryption of m_1). This is exactly the CPA security of PKE.

The κ-min-entropy of eCOM is straightforward. □

B Proof of Theorem 1

Proof. The proof mainly follows the proof in [13]. Correctness is straightforward due to the correctness of Π and the correctness of COM.

Now we prove the special soundness. Let (cmt, ch, rsp) and (cmt, ch', rsp') be two valid transcripts, where $cmt = (c_1^{(1)}, ..., c_n^{(1)}, ..., c_1^{(\ell)}, ..., c_n^{(\ell)})$, $ch := (u^{(j)})_{j\in[\ell]}$, $ch' := (u'^{(j)})_{j\in[\ell]}$, $rsp = (\mathsf{View}^{(j)}_{i\neq u^{(j)}}, d^{(j)}_{i\neq u^{(j)}})_{j\in[\ell]}$, and $rsp' = (\mathsf{View}^{(j)}_{i\neq u'^{(j)}}, d^{(j)}_{i\neq u'^{(j)}})_{j\in[\ell]}$. Moreover, there exists (at least one) $j^* \in [\ell]$ such that $u^{(j^*)} \neq u'^{(j^*)}$. The extractor $\mathcal{E}$ checks whether for $i \in [n] \setminus \{u^{(j^*)}, u'^{(j^*)}\}$, the views $\mathsf{View}_i^{(j^*)}$ are the same in both rsp and rsp'. If so, $\mathcal{E}$ retrieves the private inputs $y_i^{(j^*)}$ from views $\mathsf{View}_i^{(j^*)}$ and outputs $\bigoplus_i y_i^{(j^*)}$. Otherwise, the extractor $\mathcal{E}$ outputs $\perp$.

Thanks to the statistical binding of COM, with overwhelming probability (over the choice of the commitment key), a commitment cannot be opened into two different messages (views), and $\mathcal{E}$ does not output $\perp$. In this case, $\mathcal{E}$ is able

to collect a group of views $(\mathsf{View}_i^{(j^*)})_{i\in[n]}$, of which any two distinct views are consistent. By Lemma 1, this indicates that there exists an honest execution of Π and $\mathsf{View}_i^{(j^*)}$ is the view of P_i for all $i \in [n]$, and consequently $y_i^{(j^*)}$ contained in $\mathsf{View}_i^{(j^*)}$ are additional shares of a witness y for Y, and $\mathcal{E}$'s extraction is correct.

Finally we prove the honest-verifier zero-knowledge of $\Sigma^{\mathsf{COM}}_{\Pi,R,\ell}$. We construct a simulator Sim that outputs a simulated transcript (cmt, ch, rsp) as follows.

1. Sim independently samples $u^{(j)}$ at random for all $j \in [\ell]$.
2. For all $j \in [\ell]$, Sim randomly samples $y_i^{(j)}$ for $i \in [n]\backslash\{u^{(j)}\}$, and then invokes the simulator of the MPC protocol to obtain views $(\mathsf{View}_i^{(j)})_{i\in[n]\backslash\{u^{(j)}\}}$.
3. For all $j \in [\ell]$ and $i \in [n] \setminus \{u^{(j)}\}$, Sim commits $(\mathsf{View}_i^{(j)})_{i\in[n]\backslash\{u^{(j)}\}}$ to get $c_i^{(j)}$ and the corresponding openings $d_i^{(j)}$.
4. For all $j \in [\ell]$, Sim commits $\mathbf{0}$ (all zero bit string) to $c_{u^{(j)}}^{(j)}$.
5. Finally, Sim returns the simulated transcript (cmt, ch, rsp), where $cmt := (c_i^{(j)})_{i\in[n],j\in[\ell]}$, $ch := (u^{(1)}, ..., u^{(\ell)})$, and $rsp := ((\mathsf{View}_{i\neq u^{(j)}}^{(j)}, d_{i\neq u^{(j)}}^{(j)})_{j\in[\ell]})$.

Now we argue that the simulated transcript distributes computationally close to the transcript by an honest execution.

- The challenge simulated by Sim distributes identically to the honestly generated one since it is uniformly sampled.
- The private inputs in the response simulated by Sim distribute identically to the honestly generated ones, since they are all independent and random strings.
- For every $j \in [\ell]$, the $n-1$ views revealed in the response distribute identically (resp., statistically/computationally close) due to the $(n-1)$-privacy of the MPC protocol.
- For every $j \in [\ell]$, all commitments $c_i^{(j)}$ for $i \in [n] \setminus \{u^{(j)}\}$ are distributed identically. Moreover, the commitment $c_{u^{(j)}}^{(j)}$ will not be revealed in the transcript, and hence $c_{u^{(j)}}^{(j)}$ in the simulation distributes computationally close to the one honestly generated, due to the hiding of COM.

Therefore, Sim constructed above is an honest-verifier zero-knowledge simulator, which finishes the proof. □